Cpt S 411 Assignment Cover Sheet

Assignment #2

Jensvold, Nate

I[1] certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment. I also certify that I have not referred to online solutions that may be available on the web or sought the help of other students outside the class, in preparing my solution. I attest that the solution is my own and if evidence is found to the contrary, I understand that I will be subject to the academic dishonesty policy as outlined in the course syllabus.

Please print your names.

Nate Jensvold

Assignment Project Participant(s):
Nate Jensvold

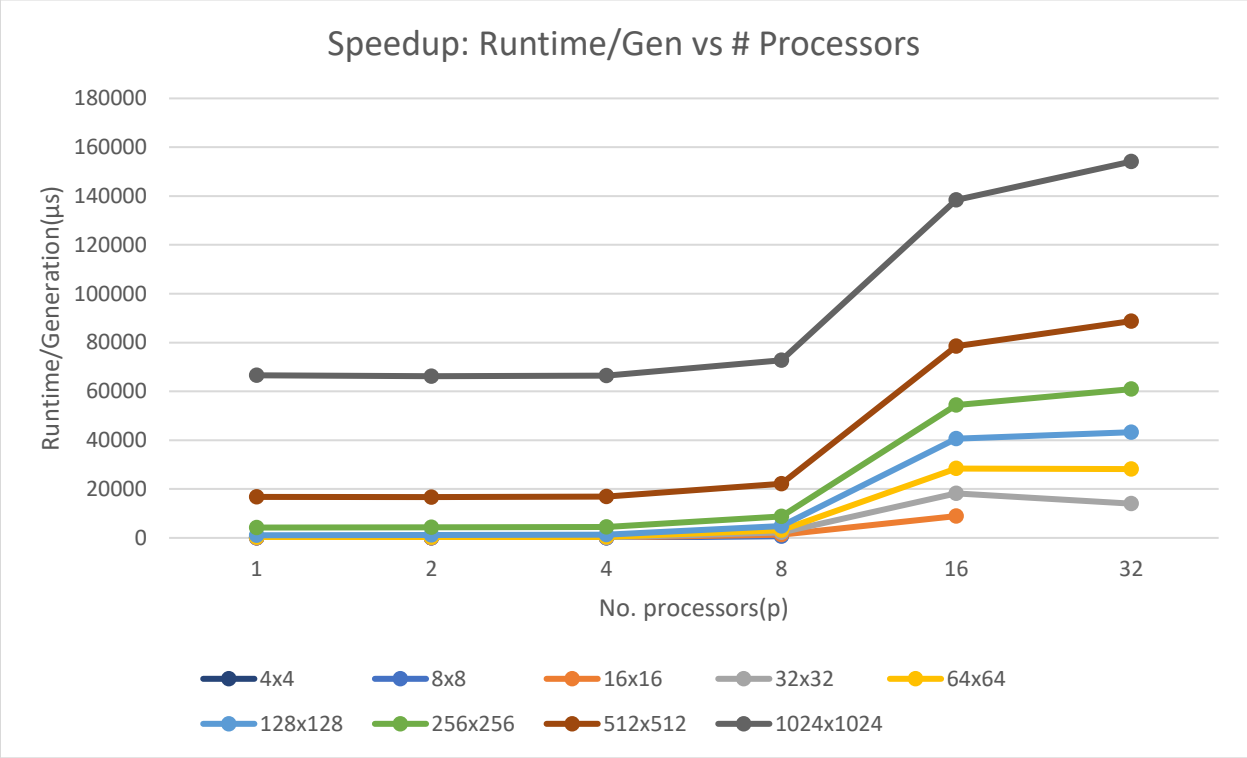
Today's Date: 10/9/2020

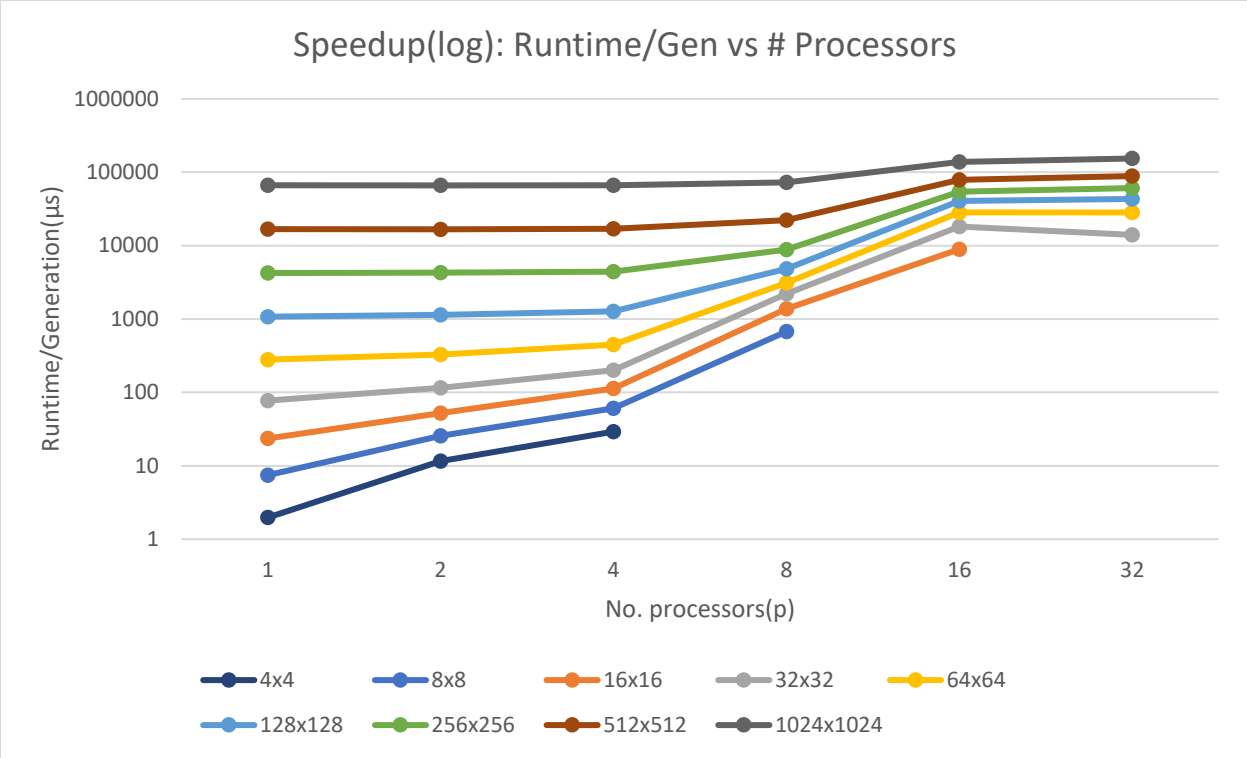
[1] If you worked as a team, then the word "I" includes yourself and your team members.

Results

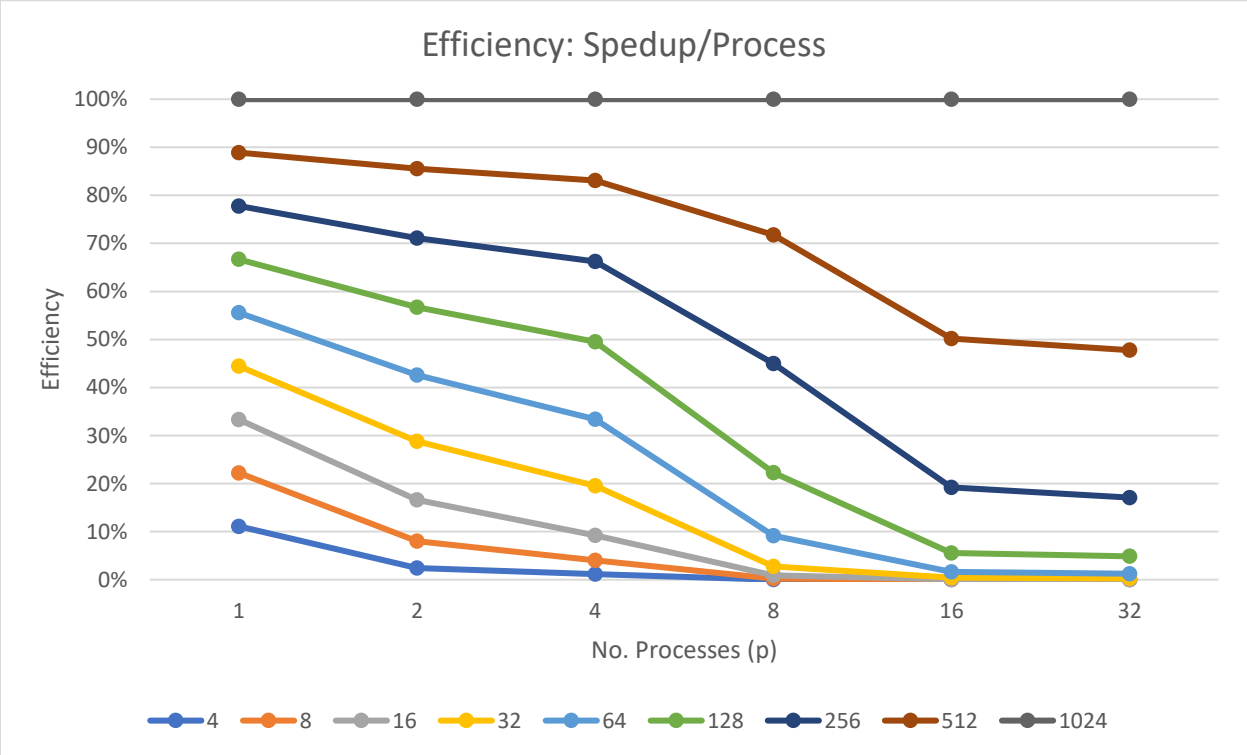
The results of this assignment were roughly in line with what I was expecting. As more processes were added the amount of time spent communicating went up significantly. The lower the number of processes the larger the percentage of total time was taken up by computation.

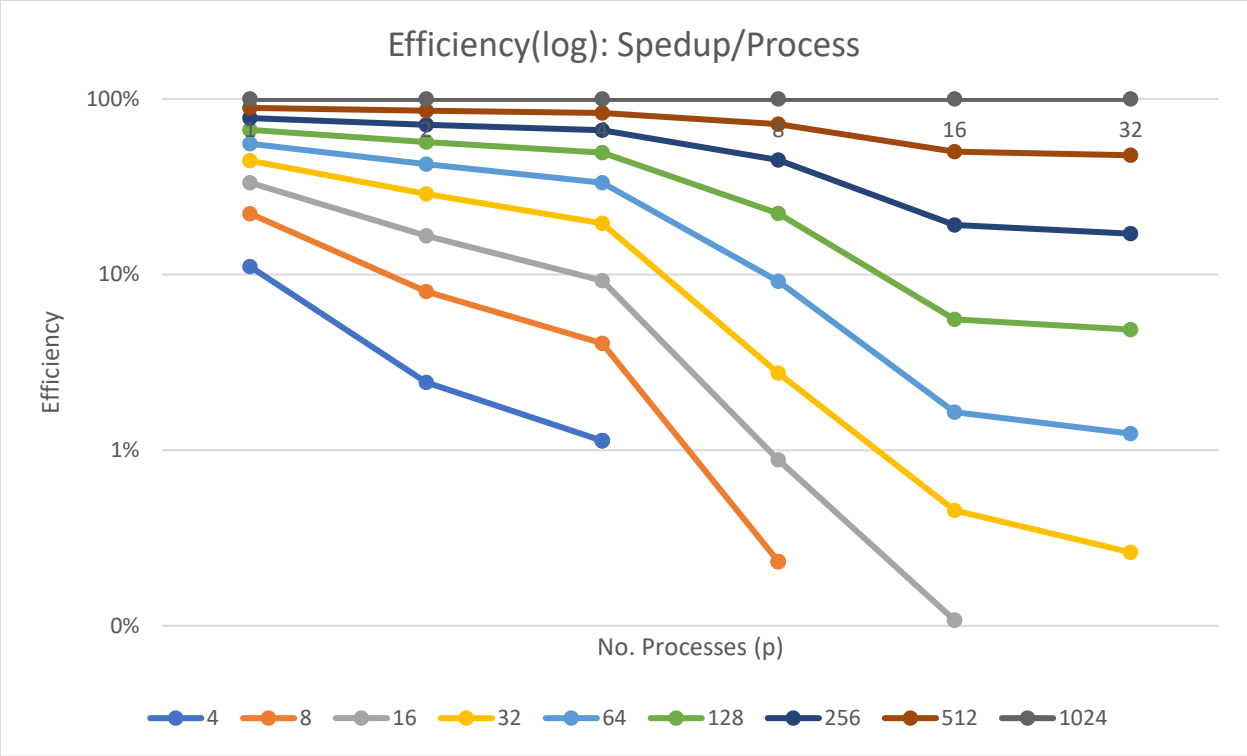
Speedup



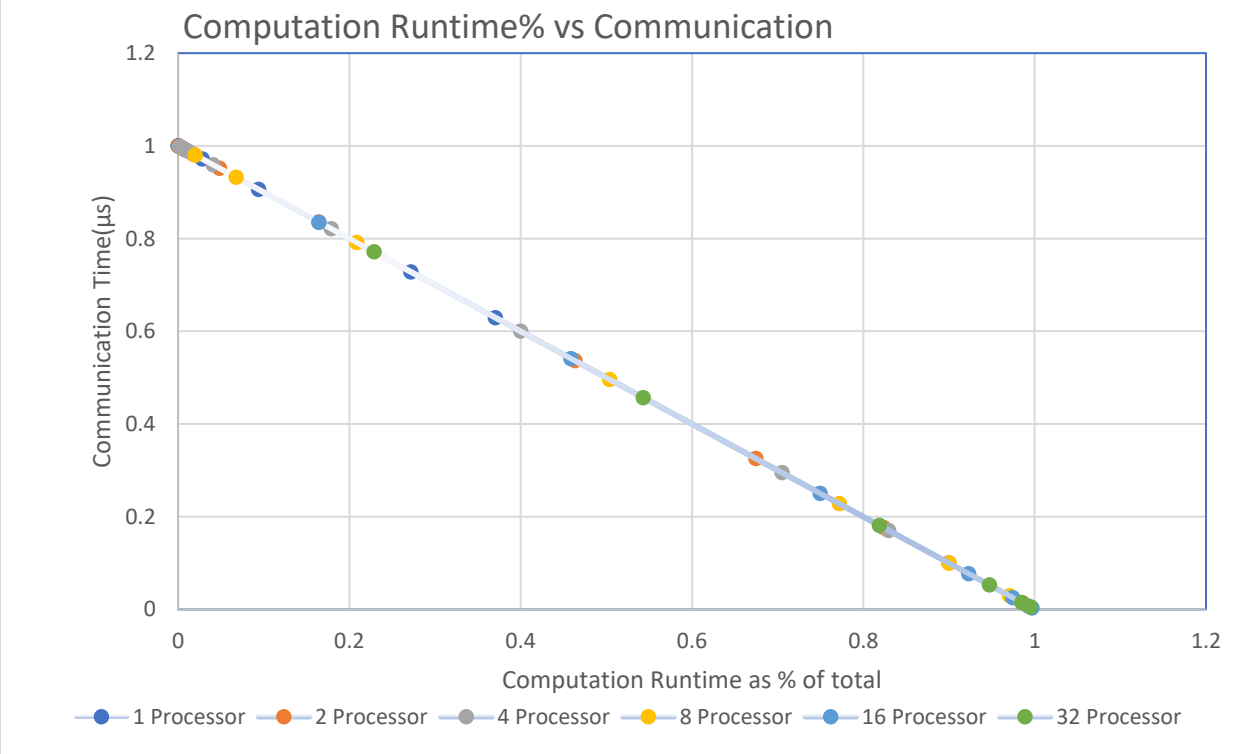

Speedup(log): Runtime/Gen vs # Processors
Runtime/Generation(µs)
1000000
100000
10000
1000
100
10
1
1
2
4
8
16
32
No. processors(p)
4x4
8x8
16x16
32x32
64x64
128x128
256x256
512x512
1024x1024

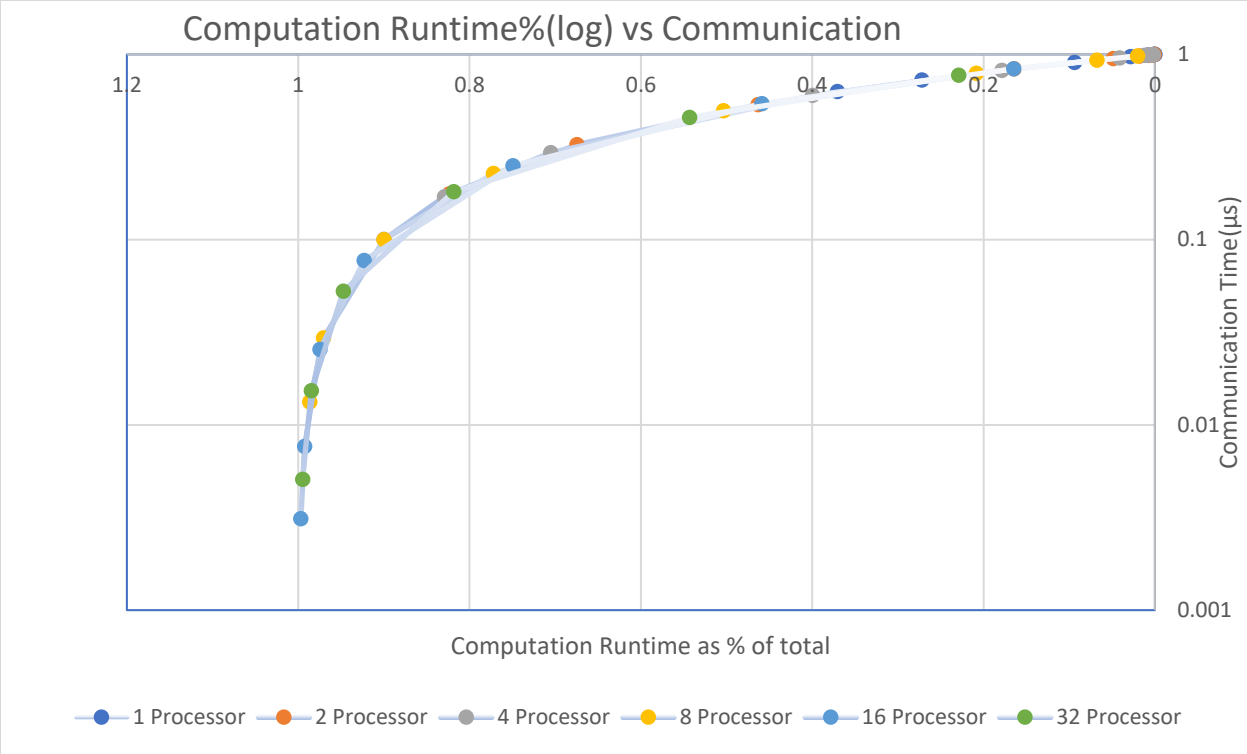
Efficiency


Efficiency: Spedup/Process
Efficiency
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
2
4
8
16
32
No. Processes (p)
4
8
16
32
64
128
256
512
1024



Computation %





Analysis

The time spent running these tests is consistent across the number of process that are used for computation. The maximum runtime for 1 process was 4991206µs at an input size of 1024x1024 while the maximum runtime for 32 process's was 6532437µs at an input size of 1024x1024.

The minimum time spent running on these tests was 197µs when using an input array of 4x4 on 1 process. The minimum time on 32 process's was 1405279µs with an input size of 32x32, additionally 1 process was able to complete 32x32 in 5384µs.

This data shows clearly that there is a significant tradeoff for introducing additional processors into this assignments "Game of life".

Average time per generation										
		Input Size								
		4	8	16	32	64	128	256	512	1024
Processes	1	1.97	5.48	16.09	53.84	202.98	797.52	3154.24	12510.15	49912.06
	2	11.63	14.1	26.72	63.46	211.19	808.9	3145.16	12434.67	49516.16
	4	29.24	31.48	52.12	87.84	246.24	829.64	3165.88	12472.79	49575.92
	8		675.71	709.39	825.71	907.76	1737.69	3984.35	13367.33	50573.62
	16			8935.4	9293.8	10162.72	12226.83	13826.7	24134.27	59847.46
	32				14052.79	14165.26	15070.37	17633.46	27872.12	65324.37

*Because of hardware issues on the cluster 64x processors were unavailable to run tests on.

Looking at the average time per generation we can see that as the number of processes increase on small input sizes the runtime per generation can increase by a factor of up to 15x. On the other end of the spectrum when the input size is large the increased processors have a smaller impact on the runtime per generation with 1 process only being ~1.3x more efficient. Looking at the runtime per generation we can see that on 1 processor it takes roughly 49912.06μs, while it takes roughly 65324μs when using 32 processors. However, one thing we have to keep in mind is that the 65324μs are a summation of all the processors times in a single generation. Because they are running concurrently the amount of time that passes per processor is actually 2041μs meaning that 32 processors are on average (49912/2041) = 24.4x faster.

This leads me to the conclusion that running this implementation of "game of life" only becomes worth using additional processes when the input size is large. At an input size of 1024x1024 the 32 process times might be 1.3x less efficient but overall, the program is 24.4x faster.